Paradise - Scalable Clusters for EOSDIS

David DeWitt & Jeff Naughton
Computer Sciences Department
University of Wisconsin

Funding: NASA (CESDIS, AISRP, EOSDIS), ARPA, IBM, Intel, SUN, Legato

Talk Outline

- Goals of the Paradise project
- Project Overview
- 1995 Activities & Current Status
- 1996 Plans

Goals of the Paradise Project

- Implement a DBMS capable of storing and processing massive geographic data sets including maps and satellite images
- DBMS for all data, not just the metadata
- Leverage ARPA-funded Gamma (parallel) and SHORE (object-oriented) DBMS technology
- Integrated supported for tertiary storage
- Target application: EOSDIS

Paradise Features

- Speed
  - innovative processing algorithms, storage structures, and parallelism
- Ease of use
  - declarative query language plus HDF support
- Scalability
  - uses scalable multiprocessor platforms and support for tertiary storage devices
- Data Integrity
  - full concurrency control and recovery services

Paradise Data Model

- Object-relational data model
- Attributes of a tuple can be instances of:
  - **standard base types**
    - int, float, string, ...
  - **predefined GIS-specific ADTs**:
    - point
    - polyline
    - polygon
  - **HDF-specific ADTs**:
    - 2 dimensional raster (8, 16, or 24 bit)
    - n-dimensional arrays (1 unlimited dimension)
  - **video (mpeg)**

Query Example

CloudCover (date: date, cloudDensity: raster16)

Cities (name: string, boundary: polygon, population: int)

```
Select name, cloudDensity.clip(boundary)
    from Cities, CloudCover
    where date = “9/15/94” and
    boundary.area() > 900 and
    cloudDensity.clip(boundary).average() > 10
```

1995 Activities

- HDF support (Kristin)
- Tertiary storage integration (JieBing)
- Parallelization (Biswadeep and Wei)
- Ports and bug fixes (Nirapuma, Kathik, Roger)

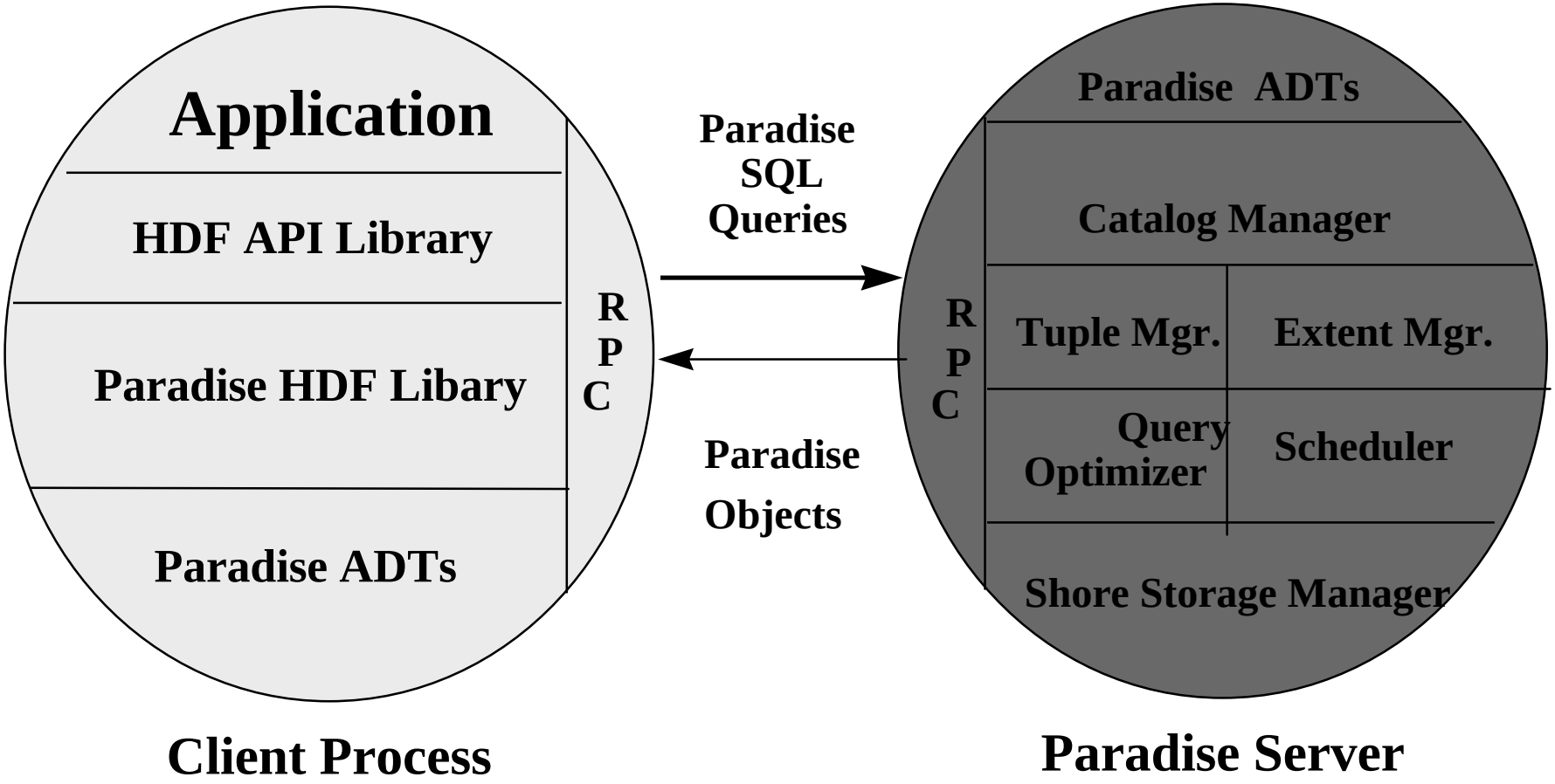
Why HDF??

- The NASA 500 don't grok SQL
- What is HDF?
  - **File format for storing scientific data sets**
    - Vdata (i.e. a table or a relation)
    - 8 bit raster image with optional color palette
    - 24 bit raster image
    - multidimensional array (with 1 unlimited dimension)
    - scale data, Vgroup (like a directory)
  - **Library interface for working with data**
    - read a hyperslab of a multidimensional array
    - "copy out" interface
  - **Reprocessing algorithms written against this interface**

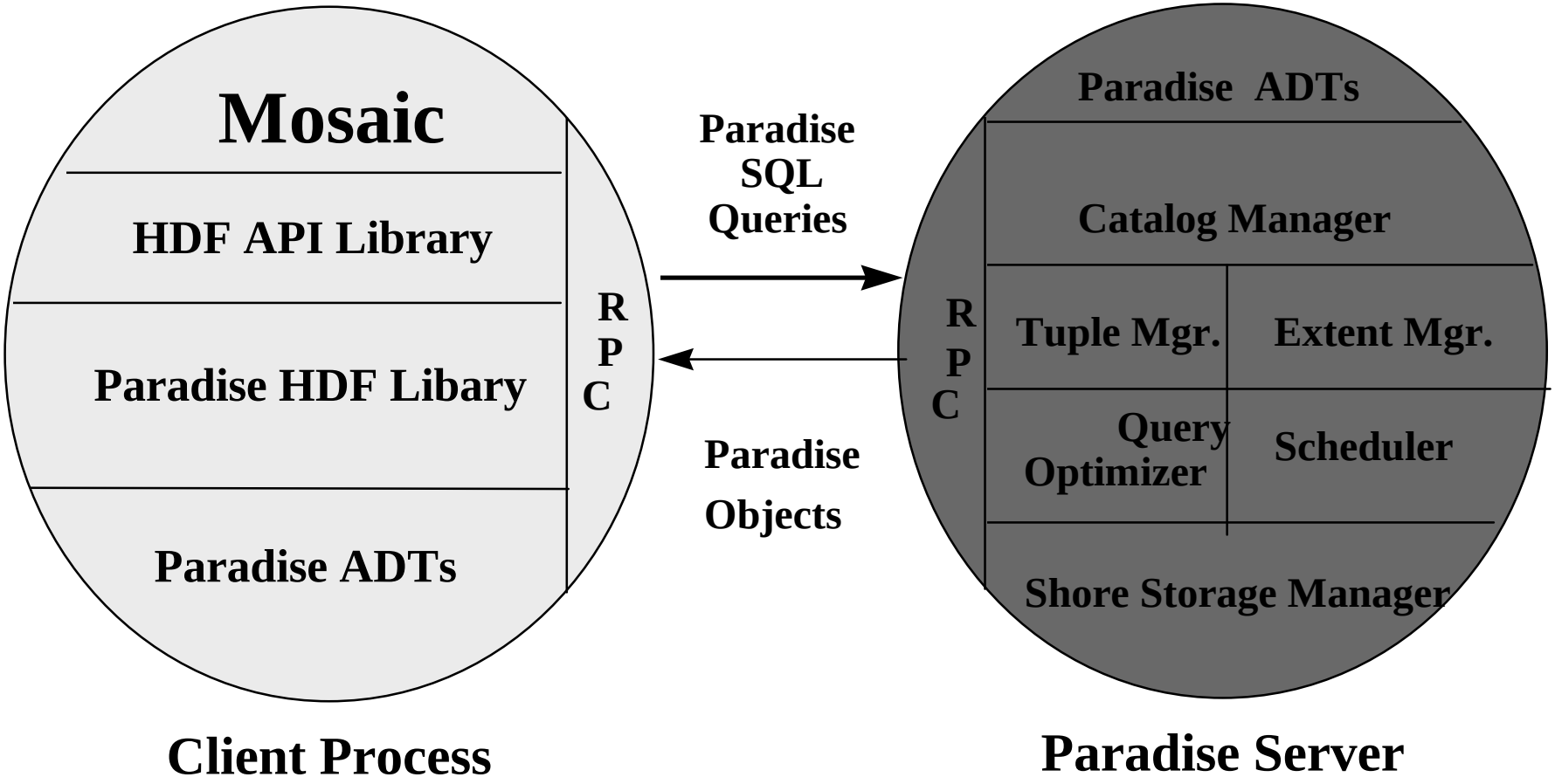
Adding HDF Support to Paradise

- Goal: provide HDF link-compatible interface
- Approach:
  - added a new ADT for each HDF data type (e.g. raster, array, ...)
  - metadata stored as normal relational attributes
  - use auxiliary “system” tables for HDF information that doesn’t map cleanly to relational data model
  - replace lower-layer of HDF library with calls to Paradise

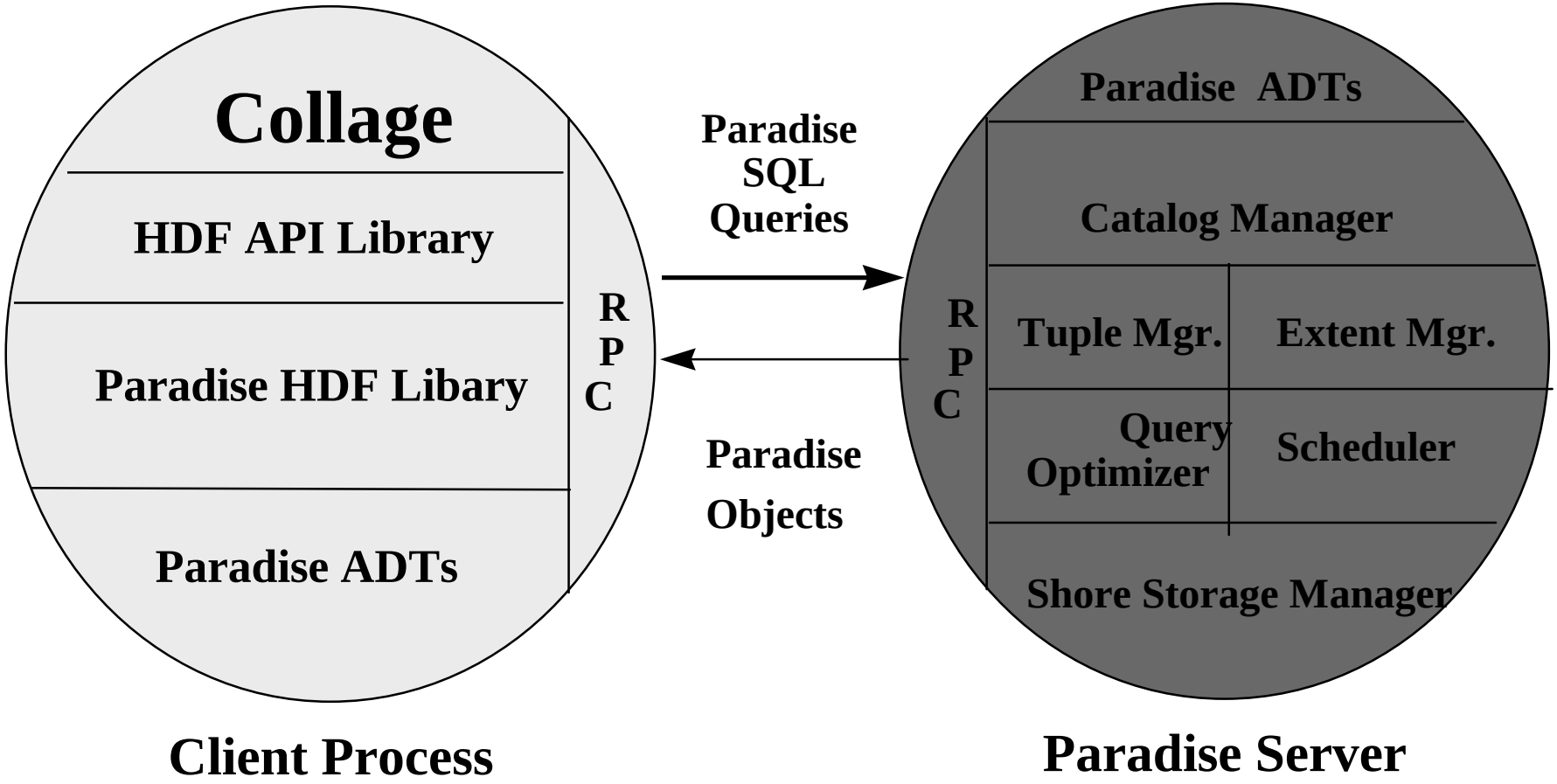
Paradise HDF Architecture



Mosaic/HDF with Paradise



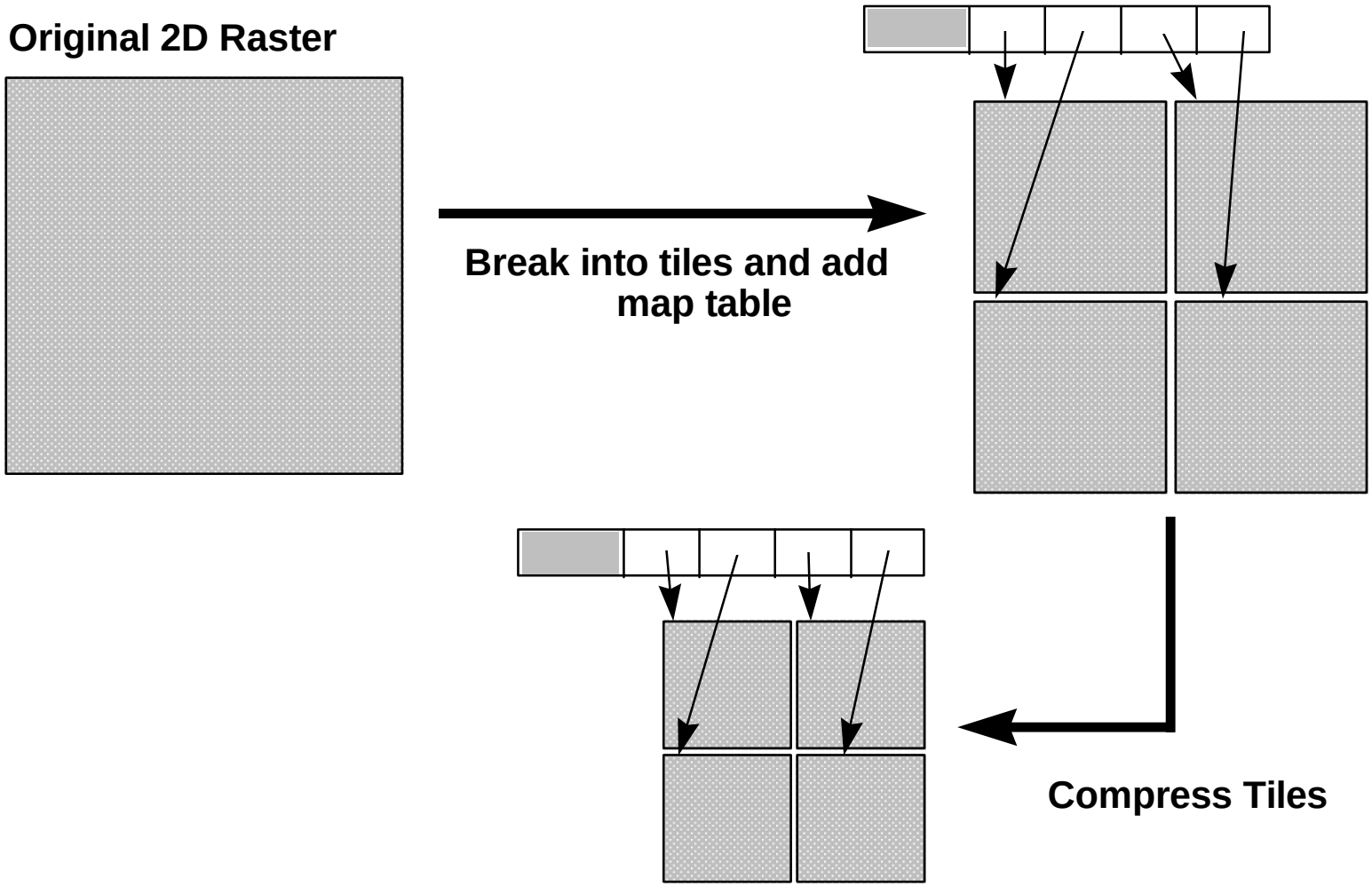
Collage with Paradise



So What?

- Why is HDF on Paradise better?
  - **HDF files limited by OS maximum file size**
  - **Paradise implementation provides chunking and compression of raster images and arrays**
  - **Integrated support for tertiary storage**
  - **Improved performance via declustering across multiple processors and/or storage devices**
  - **Can mix SQL with HDF calls**
  - **Can replace multiple HDF calls with one SQL call**

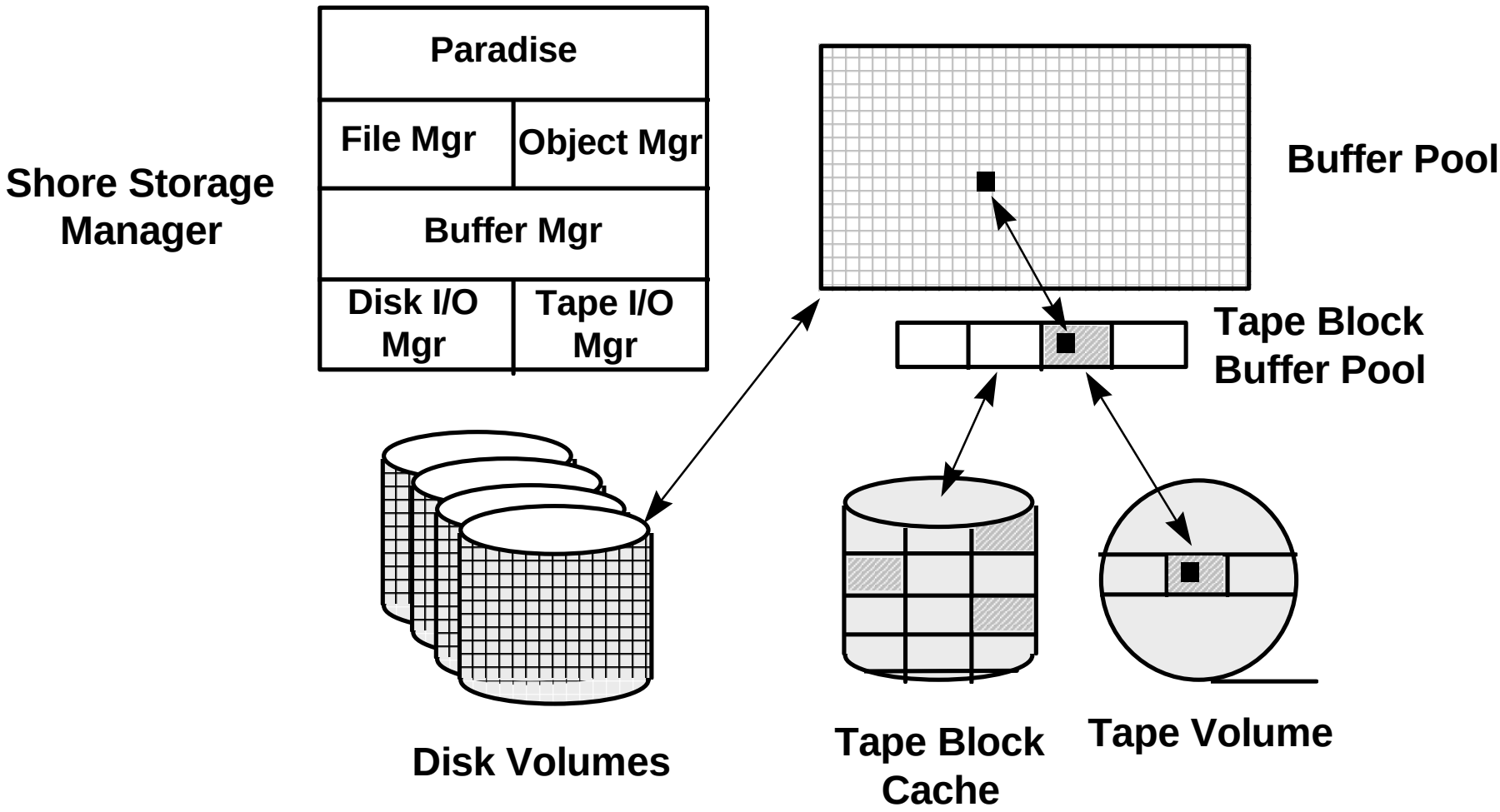
HDF Raster Implementation



Adding Tertiary Storage to Paradise

- Two Approaches:
  - **External**
    - "File" level migration (EMASS, Unitree)
    - BIG Iron
    - Application does the integration
  - **Integrated**
    - Tape becomes just another level in DBMS storage hierarchy
    - Totally transparent to application
    - Query optimizer can minimize expensive tape seeks
    - Opportunity for "query batching"

Tertiary Storage Implementation



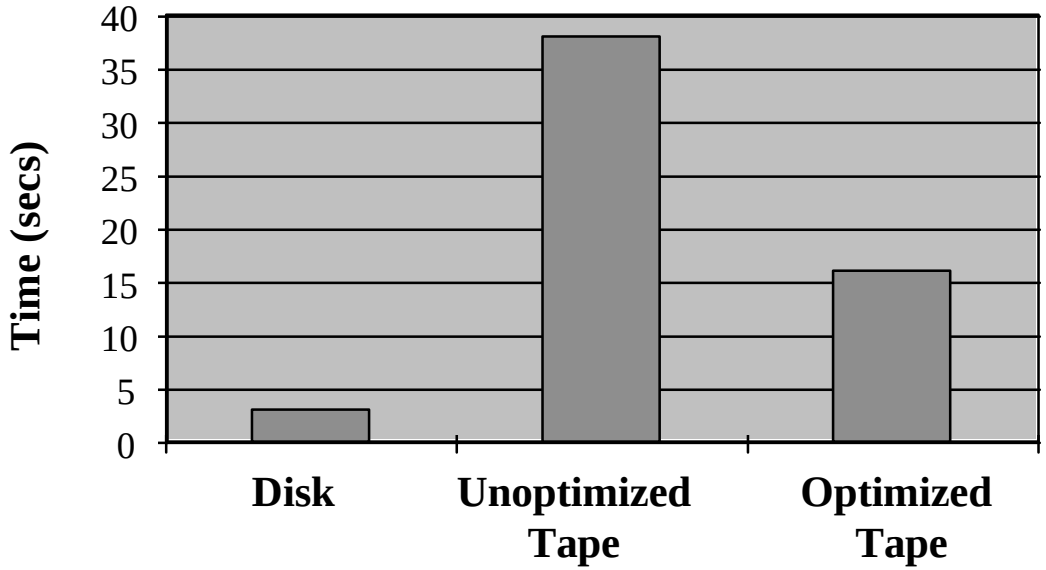
Tertiary Storage Details

- Tape and disk volumes look the same to the upper levels of the system
- Tape volume manager implements:
  - **a “log structured” file system on the tape media**
  - **disk cache for recently accessed tape blocks**
  - **mount/dismount functions for tape robot**
- Tape blocks are “big” - typically 512 Kbytes
- Query optimizer and tape volume manager reorganizes tape accesses to minimizes seeks

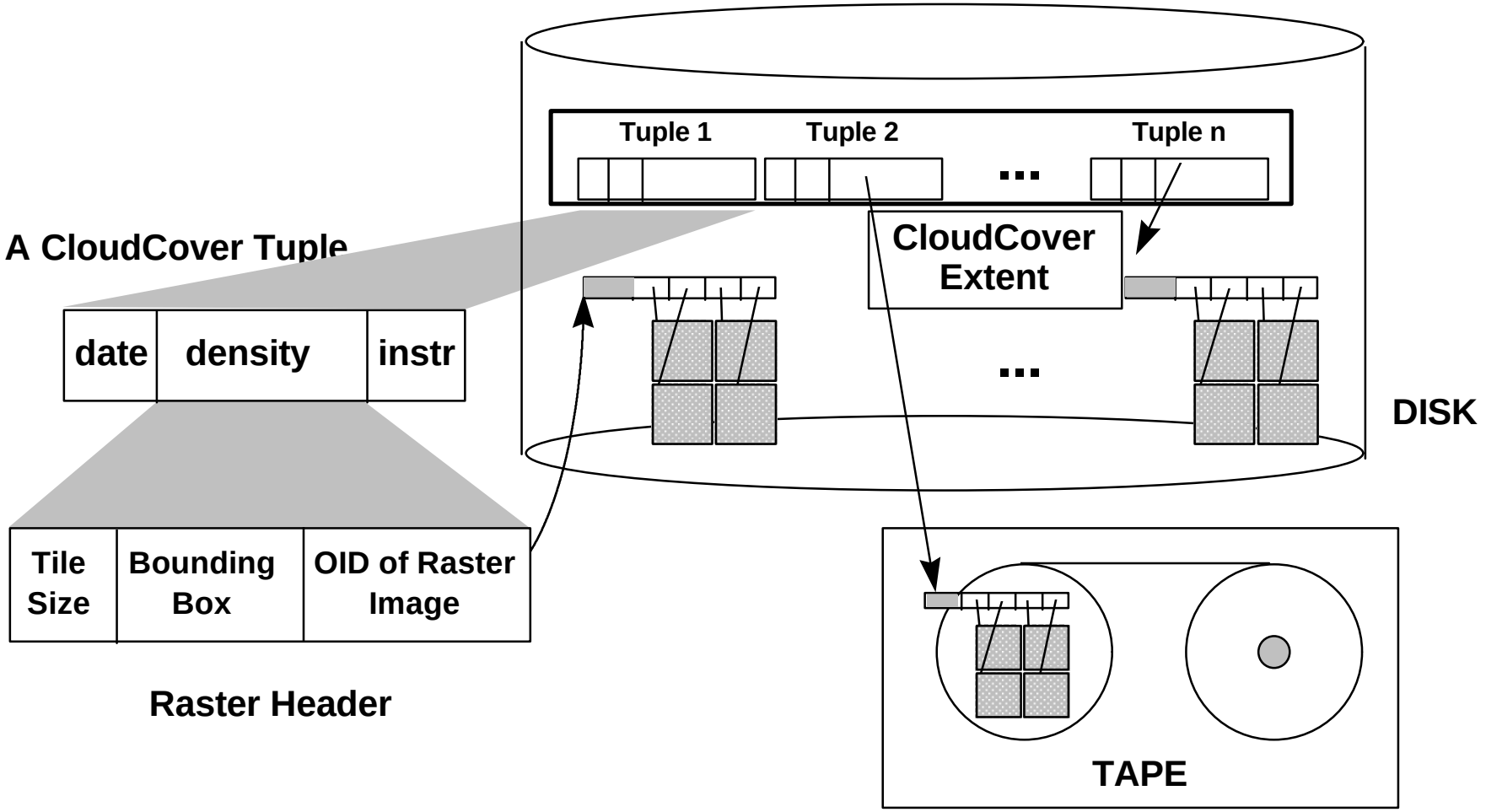
Preliminary Tape Performance

Sequoia Query #9:

```
select polygon.shape, raster.data.clip(polygon.shape)
from   polygon, raster
where  (polygon.landuse  = 92 or polygon.landuse = 91)
and    raster.freq  = 5 and  raster.time = 1
```



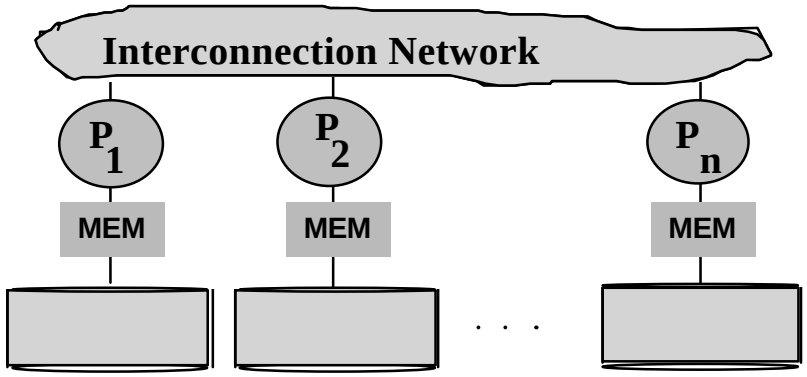
Transparency Illustration



Parallelization

- Exploit technology developed as part of Gamma project
- Scalable data archive
  - **desk top to mini-DAAC to full DAAC**
  - **“shared-nothing” design**
- Commodity hardware
  - **clusters of PCs connected via fast ethernet or ATM**
- Eliminate need for network attached storage
  - **use of commodity tertiary storage devices such as Quantum mini-robot (7 tapes @ 40 GB/tape @ 3 MB/second @ $10K)**

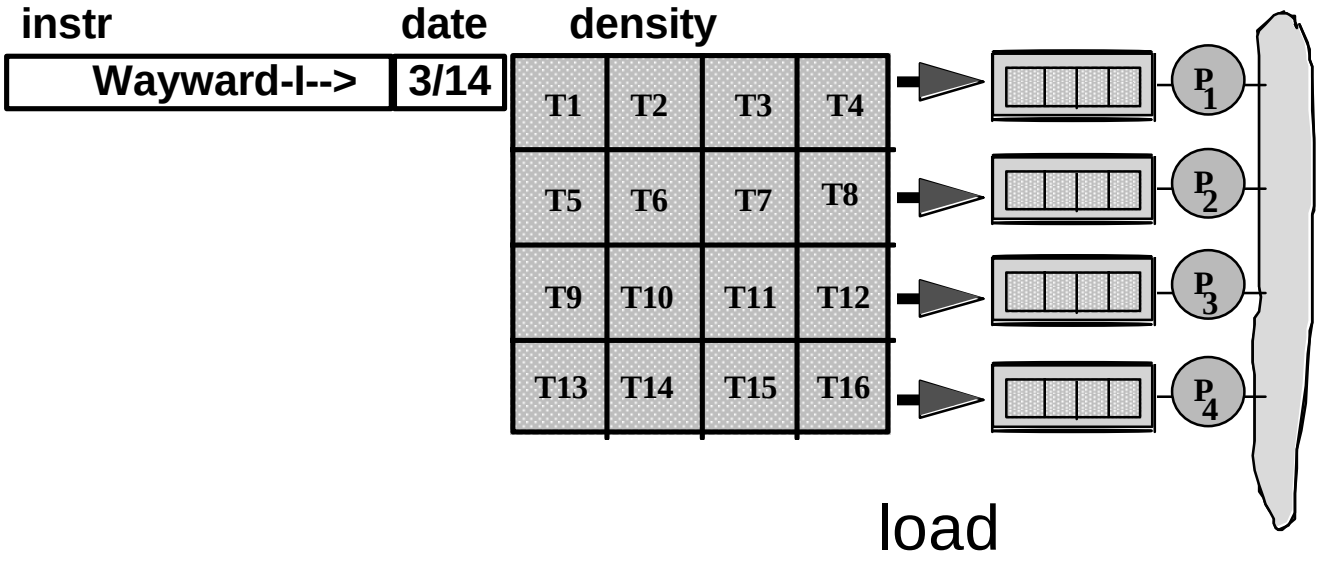
Shared-Nothing



- Scalable almost w/o limits
- Commercial examples: Teradata, NCR 3600, IBM SP2
- With ATM and fast Ethernet “roll your own” by connecting commodity workstations

Declustering Big Rasters/Arrays

CloudCover (inst: string, date: date, density: raster)



Parallelization Status

- Changes/additions required:
  - **code rewrite to do pipelining of tuples between operators**
  - **split/merge streams**
  - **scheduler and declustering mechanisms**
  - **lots of work and except for rasters/arrays and polygons, no new technology**
- Almost working!

Ports and Releases

- Ports to SGI, Solaris (PC and Sparc), & Linux completed
- Ports to NT underway (server working)
- AIX port for SP2 underway
- Releases to Goodard, CMU, SAIC, Univ. of Florida, others

1996 Plans

- Finish parallelism including rasters/arrays and polygons
- Continue query optimization for queries against tertiary storage
- Netscape front-end with Java
- Benchmarks
  - **HDF on Paradise vs. HDF on file system**
  - **Queries on Tape**
  - **“National Level” Sequoia benchmark**
- Wider distribution of code releases